CLAIMS

What is claimed is:

1. A method of a rule-based operation, comprising the steps of:

dividing a design rule into at least one of three components, the three components comprising, a design rule triggering criteria, a condition, and an action, wherein the action is responsive to an evaluation of the condition;

expressing the design rule in at least one among a datafile and source code; and

binding the three components together to form a rule object at runtime.

2. The method of claim 1, further comprising, entering at least one among a new design rule and a change to an existing design rule at runtime using a runtime plug-in, wherein the step of entering is done without compiling.

3. The method of claim 2, wherein the step of entering at least one among the new design rule and the change to the existing design rule comprises the step of changing at least a portion of one component.

4. The method of claim 1, wherein the method further comprises forming a common object interface to a rule checker program to form an interface between the three components.

5. The method of claim 1, wherein the rule-based operation is a design rule checker.

6. The method of claim 1, wherein the datafile includes an XML statement and the source code includes a C or C++ instruction.

7. A method of forming a rule object, comprising the steps of:

reading rules from at least one among a library and a textural datafile and reading rules from at least one among a dynamic rule component library and a textural rule component datafile;

constructing component objects from at least one among the library and the textural datafile and from at least one among the dynamic rule object library and the textural rule component datafile; and

matching component objects having a same identifier and revision to form the rule object.

8. The method of claim 7, wherein the method further comprises the step of binding the matching component objects that have a most recent version at runtime to form the rule object.

9. A design rule checker program, comprising:

an application criteria component for determining when a rule is triggered;

a rule condition component for determining how a positive response is tested for the rule; and

an action component for determining the action taken to detection of the positive response.

10. The design rule checker program of claim 9, wherein the program further comprises means for expressing a design rule in at least one among a datafile and source code.

11. The design rule checker program of claim 9, wherein the program further comprises means for entering at least one among a new design rule and a change to an existing design rule at runtime using a runtime plug-in.

12. The design rule checker program of claim 9, wherein the program further comprises a common object interface for interfacing with a rule object formed from the application criteria component, the rule condition component, and the action component at runtime.

13. The design rule checker program of claim 9, wherein the program further comprises a rule factory for building a rule repository for the design rule checker program from at least one among a datafile and program objects.

14. The design rule checker program of claim 13, wherein the rule factory searches specified directories for user specified rule additions.

15. A machine readable storage, having stored thereon a computer program having a plurality of code sections executable by a machine for causing the machine to perform the steps of:

dividing any design rule into at least one of three components comprising an application criteria, a rule condition, and an action;

expressing a design rule in at least one among a datafile and an object code; and

entering at least one among a new design rule and a change to an existing design rule at runtime using a runtime plug-in.

16. The machine readable storage of claim 15, wherein said step of entering is done without compiling.

17. The machine readable storage of claim 15, wherein the plurality of code sections form a common object interface to a rule checker program to form an interface between the three components.

18. The machine readable storage of claim 15, wherein the plurality of code sections bind the three components to form a rule object at runtime.

19. The machine readable storage of claim 15, wherein the computer program is a design rule checker.